UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

Instituto de Ciência e Tecnologia

Universidade Federal de São Paulo

Compiladores: Expressões regulares

Profª Thaína A. A. Tosta

tosta.thaina@unifesp.br

São José dos Campos – 2021/2

Expressões regulares

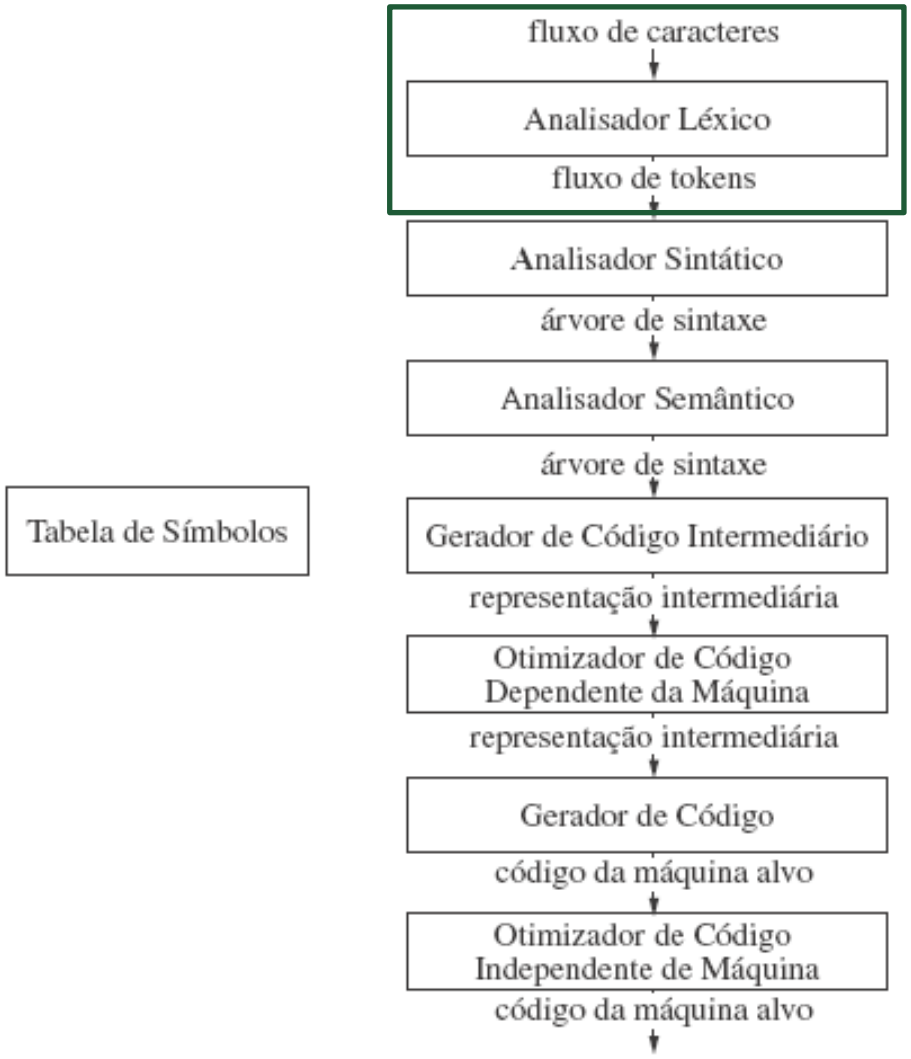


FIGURA 1.6 Fases de um compilador.

A varredura, ou análise léxica, é a fase de um compilador que lê o programa-fonte como um arquivo de caracteres e o separa em *tokens*.

Tokens	Exemplos
Palavras-chave	*if* e *while*
Identificadores	Cadeia de caracteres definida pelo usuário
Símbolos especiais	+, -, >=, <>

Expressões regulares

Por que estudar expressões regulares?

- Como a análise léxica é um tipo de casamento de padrões, temos que estudar **métodos que especifiquem e reconheçam padrões**, que são as expressões regulares e os autômatos finitos;

- Manipular o código-fonte, como faz o analisador léxico, exige cuidados para obter alta eficiência.

Expressões regulares

- No processo de análise léxica, os *tokens* válidos da linguagem normalmente são especificados por expressões regulares;
- Expressões regulares representam padrões de cadeias de caracteres;
- Uma expressão regular *r* é completamente definida pelo conjunto de cadeias de caracteres com as quais ela “casa”;
- Esse conjunto é chamado de linguagem gerada pela expressão regular, e é denotado como *L(r).*

Expressões regulares

Expressões regulares básicas

- São os caracteres em separado do alfabeto (conjunto de símbolos legais da linguagem, usualmente denotado pelo símbolo grego ∑);

- Dado um caractere *a* do alfabeto ∑ indicamos que a expressão regular **a** casa com o caractere *a* escrevendo *L(***a***) = {a}.*

Expressões regulares

Expressões regulares básicas

- Precisamos de símbolos adicionais para situações especiais
  - Utilizamos o meta-símbolo ε para denotar a cadeia vazia (uma cadeia sem caracteres)

    $L(\boldsymbol{\varepsilon}) = \{\varepsilon\}$
  - Utilizamos o meta-símbolo Φ para denotar a linguagem que não casa com nenhuma cadeia de caracteres (linguagem vazia)

    $L(\boldsymbol{\Phi}) = \{\}$

Expressões regulares

Operações básicas de expressões regulares

1. Escolha entre alternativas

- Se *r* e *s* são expressões regulares, então *r* | *s* é uma expressão regular que casa com qualquer cadeia que case com *r* ou com *s;*
- Em termos de linguagem, a linguagem de *r* | *s* é a **união** de *r* e *s*

$$L(\mathbf{r}|\mathbf{s}) = L(\mathbf{r}) \cup L(\mathbf{s}).$$

Exemplos:

Para a expressão regular ***a*** **|** ***b***:

$$L(\mathbf{a}|\mathbf{b}) = L(\mathbf{a}) \cup L(\mathbf{b}) = \{a\} \cup \{b\} = \{a, b\}$$

Para a expressão regular ***a*** **|** ***ε***:

$$L(\mathbf{a}|\boldsymbol{\varepsilon}) = \{a, \varepsilon\}$$

Para a expressão regular ***a*** **|** ***b*** **|** ***c*** **|** ***d***:

$$L(\mathbf{a}|\mathbf{b}|\mathbf{c}|\mathbf{d}) = \{a, b, c, d\}$$

Expressões regulares

Operações básicas de expressões regulares

2. Concatenação

A concatenação de duas expressões regulares *r* e *s* é denotada por *rs*, e casa com qualquer cadeia de caracteres que seja a concatenação de duas cadeias, desde que a 1ª case com *r* e a 2ª case com *s.*

*L(**rs**) = L(**r**)L(**s**)*

Exemplos:

Para a expressão regular **(a|b)c**,

*L(**(a|b)c**) = L(**a|b**)L(**c**) = {a,b}{c} = {ac, bc}*

Para a expressão regular *rs*, em que *r = (a|b), s = (c|d), temos*

*L(**rs**) = L(**(a|b)(c|d)**) = L(**a|b**)L(**c|d**) = {a, b}{c, d} = {ac, ad, bc, bd}*

Expressões regulares

Operações básicas de expressões regulares

3. Repetição

- Seja uma expressão regular *r*, a operação de repetição é denotada por *r**, sendo também chamada de fecho (de *Kleene*)
- A expressão *r** casa com qualquer ***concatenação*** finita de cadeias de caracteres (inclusive a cadeia vazia), desde que cada cadeia case com *r*

  Seja a expressão regular a*,

  L(a*) = L(a)* = {a}* = {ε, a, aa, aaa, ...}

  Exemplo:

  Considere a expressão regular (a|bb)*

  L(**(a|bb)***) = L(**a|bb**)* = {a, bb}* = {ε, a, bb, aa, abb, bba, bbbb, aaa, aabb, abba, bbaa, ...}

Expressões regulares

Operações básicas de expressões regulares

- Precedência de operações
  1. Repetição
  2. Concatenação
  3. Escolha (alternativas)
- Uso de parênteses
  - Quando queremos indicar uma precedência diferente, devemos utilizar parênteses
  - Exemplos:

    (a|b)c a escolha terá precedência sobre a concatenação

    (a|b)* a escolha terá precedência sobre a repetição

Expressões regulares

Operações básicas de expressões regulares

Nomes para expressões regulares

É útil simplificar a notação com nomes significativos para expressões regulares.

Exemplo:

dígito dígito*

onde

dígito = 0|1|2|...|9

dizemos que o nome *dígito* é uma definição regular

Expressões regulares

Definição de expressão regular

1. Uma expressão regular básica, composta por um único caractere **a** (onde *a* pertence a um alfabeto ∑ de caracteres legais), o metacaractere ε ou o metacaractere Φ
   - No 1º caso, L(**a**) = {a}
   - No 2º caso L(**ε**) = {ε}
   - No 3º caso L(**Φ**} = {}
2. Uma expressão da forma *r*|*s*, onde *r* e *s* são expressões regulares

   *L(**r**|**s**) = L(**r**) U L(**s**)*
3. Uma expressão da forma *rs*, onde *r* e *s* são expressões regulares

   *L(**rs**) = L(**r**)L(**s**)*

Expressões regulares

Definição de expressão regular

4. Uma expressão da forma *r**, onde *r* é uma expressão regular

   $L(r^*) = L(r)^*$

5. Uma expressão da forma *(r)*, onde *r* é uma expressão regular

   $L((r)) = L(r)$

Os parênteses são utilizados apenas para ajustar a precedência dos operadores.

Expressões regulares

Operações estendidas de Expressões Regulares

- Uma ou mais repetições: seja *r* uma expressão regular, *r+* indica **uma** ou mais repetições de *r*

  Exemplo: (0|1) (0|1)* = (0|1)+

- Intervalo de caracteres: utiliza-se os metacaracteres colchetes e hífen

  Exemplos: [a – z] [0 – 9] [a-zA-Z] [abc]

  Essa notação com colchetes é chamada de **classe de caracteres**

- Qualquer caractere fora de um conjunto: utiliza-se os metacaracteres til (~) ou circunflexo (^), que indica "não" ou o complemento

  Exemplos: *~(a|b|c) ou [^abc]* qualquer caractere que não seja a, b ou c

  [^a] expressão regular para um caractere no alfabeto que não seja *a*

Expressões regulares

Operações estendidas de Expressões Regulares

- Subexpressões opcionais: utiliza-se o metacaractere interrogação (?), com *r* sendo uma expressão regular. Assim, *r?* indica que as cadeias que casam com *r* são opcionais

  Exemplo: *natural* = [0-9]+

  naturalComSinal = (+|-)? *natural*

- Qualquer caractere: utiliza-se o metacaractere ponto "." para representar o casamento com qualquer caractere do alfabeto

  Exemplo: todas as cadeias de caractere que contêm pelo menos um *m*

  *.*m.**

Expressões regulares

Operações estendidas de Expressões Regulares

- Caractere de escape: utiliza-se o metacaractere barra invertida "\" seguido do caractere literal que pretende-se especificar. Assim, o caractere de escape permite que um metacaractere seja interpretado de forma literal

  Exemplo: r = [0-9]\.[0-9]

  *L(**r**) = {0.0, 0.1, 0.2..., 1.0, 1.1, 1.2...}*

Testador de expressão regular

Regex Tester: https://regex101.com

Expressões regulares

Ambiguidade

- Ocorre quando uma cadeia pode casar com mais do que uma expressão regular;
- Podemos resolver esse problema pelo uso de palavras reservadas, para as quais deve haver uma expressão regular para cada

  Exemplos*: if else do while*
- Uma segunda alternativa é pelo princípio da subcadeia mais longa, em que a expressão regular que casa com a cadeia mais longa deve ser adotada

  Exemplos: **>= == <= <> !=**

Expressões regulares

Delimitadores de lexemas

- Espaço em branco;
- Caractere de tabulação;
- Caractere de mudança de linha;
- Caracteres que não casam com a expressão regular em análise;

Exemplos:

x = 10 (delimitados por espaços em branco)

a=b+c

Expressões regulares

Expressões regulares para marcas de linguagem de programação

As linguagens de programação utilizam lexemas que se enquadram em categorias limitadas e padronizadas, como palavras reservadas, símbolos especiais, identificadores, constantes e literais.

Números: natural = [0-9]+
naturalComSinal = (+|-)? natural
número = naturalComSinal(\.natural)?

Palavras reservadas: reservadas = if | while | do ...

Identificadores: letra = [a-zA-Z]
dígito = [0-9]
identificador = letra(letra|dígito)*

Expressões regulares

Apresente o conjunto de cadeias de caracteres (com pelo menos 5 elementos, quando couber) que casam com as expressões regulares

a) a

*L(**a**) = {a}*

b) a*

*L(**a***) = L(a)* = {ε, a, aa, aaa, aaaa, ...}*

c) a+

*L(**a+**) = L(a)+ = {a, aa, aaa, aaaa, aaaaa, ...}*

d) (a|b|c)+

e) (ab)*

f) (0|1)+

g) ab*(c|ε)

h) (ab)*(cd)*

Expressões regulares

Apresente o conjunto de cadeias de caracteres (com pelo menos 5 elementos, quando couber) que casam com as expressões regulares

a) a

*L(**a**) = {a}*

b) a*

*L(**a***) = L(a)* = {ε, a, aa, aaa, aaaa, ...}*

c) a+

*L(**a+**) = L(a)+ = {a, aa, aaa, aaaa, aaaaa, ...}*

d) (a|b|c)+

*L(**(a|b|c)+**) = {a, b, c, aa, ab, bc, abc, aabbcc, ...}*

e) (ab)*

*L(**(ab)***) = {ε, ab, abab, ababab, abababab, ...}*

f) (0|1)+

*L(**(0|1)+**) = {0, 1, 01, 11, 011 ...}*

g) ab*(c|ε)

*L(**ab*(c|ε)**) = {a, ab, ac, abb, abc, ...}*

h) (ab)*(cd)*

*L(**(ab)*(cd)***) = {ε , ab, cd, abab, cdcd, abcd, ...}*

Expressões regulares

A partir das cadeias de caracteres apresentadas, encontre as expressões regulares que as definem:

a) *{a, b, ab, abba, aaab, baaa, ...}*

b) *{abc, aabc, aaabc, aaaabc, ...}*

c) *{1, 2, 20, 1234, 56789, 803459, ...}*

d) *{if, else, do, while}*

e) *{casa, nota, media, Peso, zap, IRPF, ...}*

f) *{nota01, p1, xyz, xyz123, cod2f1,...}*

g) *números hexadecimais em linguagem C*

h) números reais

Expressões regulares

A partir das cadeias de caracteres apresentadas, encontre as expressões regulares que as definem:

a) *{a, b, ab, abba, aaab, baaa, ...}* ***(a|b)+***

b) *{abc, aabc, aaabc, aaaabc, ...}* ***a*abc***

c) *{1, 2, 20, 1234, 56789, 803459, ...}* ***[0-9]+***

d) *{if, else, do, while}* ***if | else | do | while***

e) *{casa, nota, media, Peso, zap, IRPF, ...}* ***[a-zA-Z]+***

f) *{nota01, p1, xyz, xyz123, cod2f1,...}* ***[a-z]([a-z]|[0-9])****

g) *números hexadecimais em linguagem C* ***0x([0-9]|[A-F])+***

h) números reais ***(+|-)?[0-9]+(\.[0-9]+)?***

Expressões regulares

Considere o alfabeto ∑ = {a,b,c,d...,z}

a) Apresente uma expressão regular que aceite cadeias de caracteres que contêm exatamente um z

[^z]*z[^z]*

b) Apresente uma expressão regular que aceite cadeias de caracteres que contêm no máximo um z

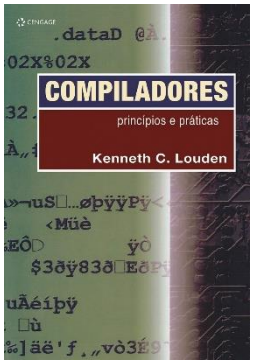
[^z]*[z|ε][^z] *ou* ***[^z]*z?[^z]****

Considere o alfabeto ∑ = {a,b,c}, e apresente uma expressão regular que aceite cadeias que não contêm dois b's consecutivos

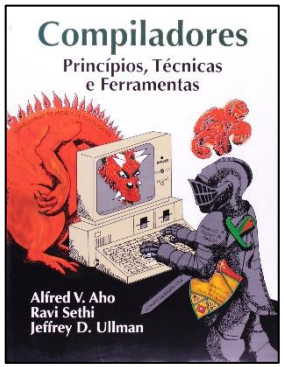
(a|c|ba|bc)*(b|ε)

Expressões regulares

Bibliografia consultada:



LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004 (Cap. 2);



AHO, A. V.; LAM, M. S.; SETHI, R. e ULLMAN, J. D. **Compiladores: princípios, técnicas e ferramentas.** 2ª edição – São Paulo: Pearson Addison-Wesley, 2008 (Cap. 3).

Agradeço ao Professor Luiz Eduardo G. Martins pelo material disponibilizado.